OULUN SEUDUN
AMMATTIKORKEAKOULU

Hannu Kyllönen

HATTRICK-PELIN ASIAKASOHJELMAN EDELLEENKEHITTÄMINEN

HATTRICK-PELIN ASIAKASOHJELMAN EDELLEENKEHITTÄMINEN

Hannu Kyllönen
Opinnäytetyö
Syksy 2011
Tietotekniikan koulutusohjelma
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, ohjelmistojen kehitys

Tekijä(t): Hannu Kyllönen
Opinnäytetyön nimi: Hattrick-pelin asiakasohjelman edelleenkehittäminen
Työn ohjaaja(t): Kari Laitinen
Työn valmistumislukukausi ja -vuosi: Syksy 2011 Sivumäärä: 32

Tämän opinnäytetyön tarkoituksena oli edelleenkehittää aiemmin harrasteprojektina aloitettua Android-sovellusta. Sovelluksen tarkoituksena on tuoda osa Hattrick-pelin ominaisuuksista pelattavaksi mobiililaitteessa. Hattrick-peli on verkkoselaimella pelattava jalkapallomanageripeli, jossa pelaajalla on oma virtuaalinen jalkapallojoukkue.

Työssä sovellukseen lisättiin pelaajanäkymä ja tietokanta sekä optimoitiin käyttöliittymää. Pelaajanäkymä tuo sovellukseen tärkeän ominaisuuden, jolla käyttäjä voi selata omistamiaan pelaajia ja niiden tietoja. Tiedonsiirron vähentämiseksi sovellukseen lisättiin tietokanta, jolloin käyttäjän jo lataamia tietoja voidaan käyttää uudelleen. Paremman käyttökokemuksen saamiseksi sovelluksen käyttöliittymää optimoitiin.

Työn tuloksena sovellukseen saatiin halutut lisäominaisuudet sekä käyttömukavuus parani huomattavasti.

Asiasanat: Android, sovelluskehitys, Hattrick

SISÄLLYS

1 JOHDANTO

Tämä opinnäytetyö käsittelee keväällä 2011 alkanutta harrasteprojektia, jonka tavoitteena oli kehittää sovellus Hattrick-nimisen pelin pelaamiseen Android-käyttöjärjestelmää käyttävissä mobiililaitteissa. Hattrick-peli on verkkoselaimella pelattava jalkapallomanageripeli, jossa jokaisella pelaajalla on oma virtuaalinen jalkapallojoukkue.

Luvussa 2 käydään läpi Android-käyttöjärjestelmää yleisellä tasolla. Aluksi tutustutaan käyttöjärjestelmän historiaan, jonka jälkeen siirrytään käyttöjärjestelmän arkkitehtuuriin. Lopuksi esitellään vielä muutamia yleisesti käytettyjä sovelluskomponenttia.

Luvussa 3 tutustutaan hieman Hattrick-peliin. Neljännessä luvussa käydään läpi projektin tuotoksena syntynyttä HatMob-sovellusta. Luvussa esitellään sovelluksen yleistä rakennetta, käyttöliittymää ja sovellusmoottoria.

Viidennessä luvussa otetaan tarkasteluun sovellukseen kehitetty pelaajanäkymä ja tietokanta sekä käydään läpi käyttöliittymän optimointia. Pelaajanäkymä tuo sovellukseen ominaisuuden, jolla käyttäjä pääsee selaamaan oman joukkueensa pelaajia. Tietokannan tarkoituksena on vähentää tiedonsiirtoa palvelimen ja mobiililaitteen välillä. Myöhemmin talletettuja tietoja voidaan käyttää esimerkiksi pelaajan taitokehitysten seuraamiseen. Hyvän käyttäjäkokemuksen saavuttamiseksi optimoitiin sovelluksen käyttöliittymää.

2 ANDROID-TEKNOLOGIA

2.1 Historia

Android-alustan kehittämisen aloitti vuonna 2003 perustettu yhdysvaltalainen Android Inc. -yhtiö. Yhtiön perustajiin kuului Andy Rubin, Rich Miner, Nick Sears ja Chris White. Yhtiö kehitti Androidia vuoteen 2005 asti, jolloin Google osti yrityksen. Kaupan jälkeen pääkehittäjistä Andy Rubin, Rich Miner ja Chris White jatkoivat kehitystä Googlen alaisuudessa. Kaupan myötä syntyi huhuja siitä, että Google olisi laajentamassa toimintaansa mobiilimarkkinoille. Linuxin ytimeen pohjautuva alusta kehittyi nopeasti Andy Rubinin johdolla. Google markkinoi alustaa joustavana ja helposti päivitettävänä. (1.)

Vuonna 2008 HTC julkaisi ensimmäisen Android-käyttöjärjestelmään perustuvan puhelimen. HTC Dream -puhelin oli varustettuna Android 1.0 -käyttöjärjestelmällä.

Tuolloin käyttöjärjestelmä sisälsi paljon älypuhelimen perusominaisuuksia, kuten selaimen, sähköpostin, kalenterin, mediasoittimen, navigaattorin ja kameran. Vuonna 2009 julkaistiin versio 1.5 (kuva 1), joka toi mukanaan lukuisia ominaisuuksia ja käyttöliittymäpäivityksiä. 1.5-version myötä otettiin myös käyttöön versioiden koodinimet, joita ovat Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb ja viimeisimpänä Ice Cream Sandwich. Kuvassa yksi esitetään Android 1.5 -version kotinäkymä. (2.)

KUVA 1. Android 1.5:n kotinäkymä (2)

2.2 Käyttöjärjestelmän arkkitehtuuri

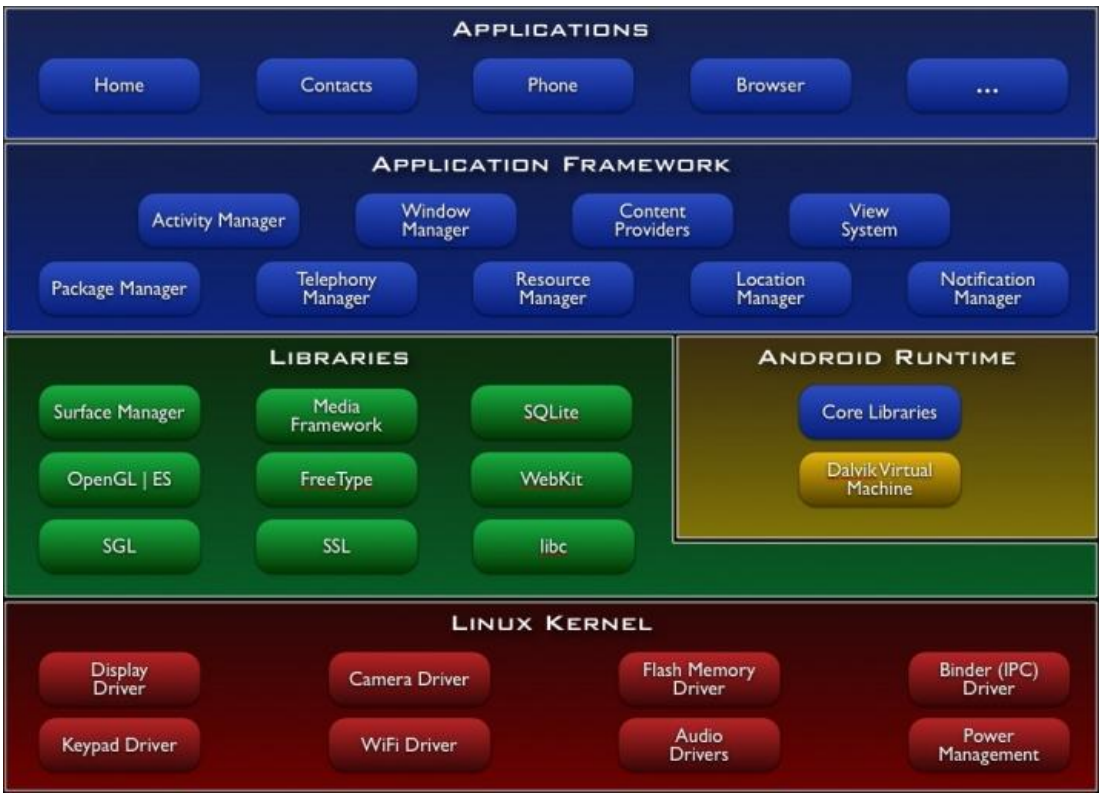
Käyttöjärjestelmän arkkitehtuuri voidaan jakaa viiteen eri osaan (kuva 2). Alimmalla tasolla on Linuxiin pohjautuva ydin, joka tarjoaa käyttöjärjestelmän peruspalvelut, kuten muistinhallinnan, prosessienhallinnan, tietoturvan ja tietoliikenteen. Ytimen tehtävänä on myös tarjota rajapinta laitteistoon. (3.)

Seuraavalla tasolla ovat Dalvik-virtuaalikone sekä käyttöjärjestelmän ytimen kirjastot. Virtuaalikone mahdollistaa Java-sovellusten suorittamisen Android-laitteissa (3).

Android sisältää lukuisia C- ja C++-kirjastoja, joita useat eri käyttöjärjestelmän komponentit käyttävät. Esimerkkinä voidaan ottaa mediakirjastot, jotka tuovat mahdollisuuden toistaa ja nauhoittaa useita tunnettuja ääni- ja videoformaatteja sekä näyttää kuvia. (3.)

Neljännellä tasolla on sovelluskehys. Se tarjoaa käyttöjärjestelmässä suoritettaville sovelluksille rungon, jonka ansiosta sovelluskehitys käyttöjärjestelmälle on helpompaa (3).

Viimeisellä, eli ylimmällä tasolla ovat käyttöjärjestelmän sovellukset. Niitä ovat esimerkiksi selain, kontaktit ja puheluiden soittamiseen käytetty sovellus (3).

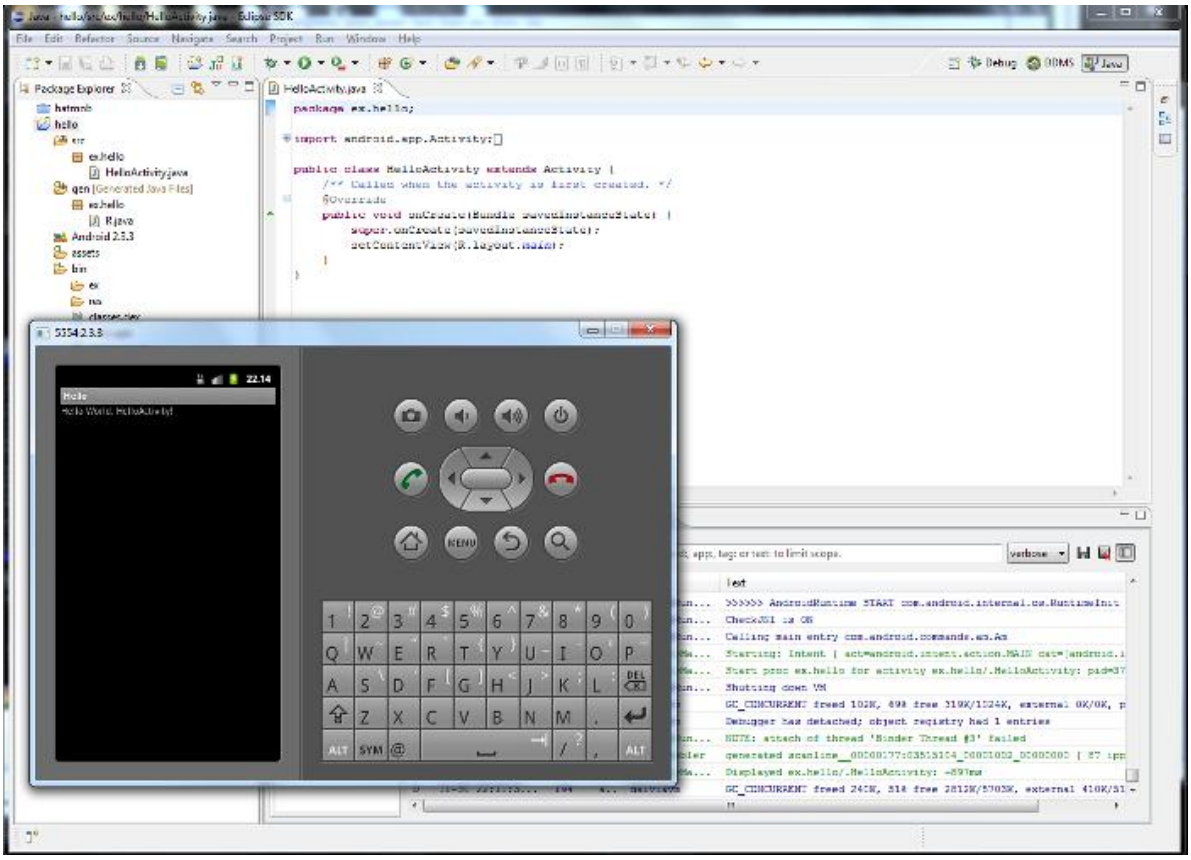


KUVA 2. Android-käyttöjärjestelmän arkkitehtuuri (3)

2.3 Ohjelmistokehitys

Ohjelmien kehitys Android-alustalle tapahtuu pääasiassa Java-ohjelmointikielellä ja tarjolla olevien työkalujen avulla (Android Software Development Kit). Alustalle on myös mahdollista tehdä kirjastoja esimerkiksi C- ja C++-ohjelmointikielillä. (4.)

Ohjelmistokehitys tapahtuu yleisesti Eclipse-kehitysympäristössä (kuva 3). Android-työkaluja voidaan kuitenkin käyttää myös niin sanotusti komentorivipohjaisesti, jolloin kehittäjää ei ole pakotettu minkään tietyn ympäristön käyttöön. (4.)



KUVA 3. Android SDK Eclipse -kehitysympäristössä

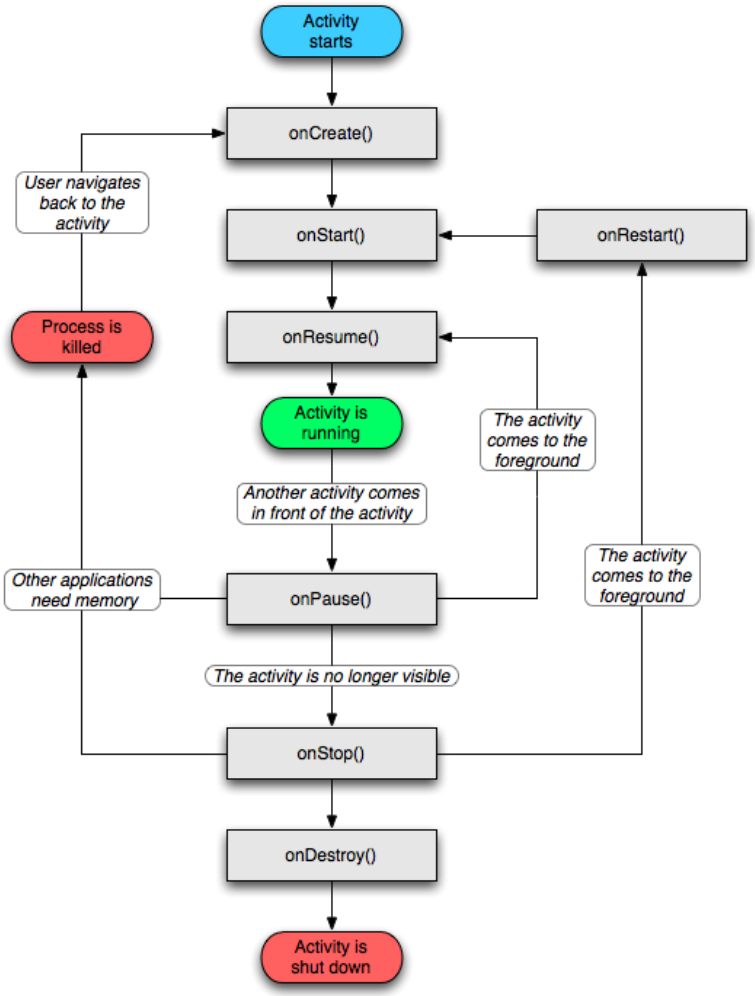
2.4 Androidin peruskomponentit

Activity

Aktiviteetti (Activity) on komponentti, joka tarjoaa käyttäjälle ohjelman näkymän ja jonka kautta käyttäjä voi olla vuorovaikutuksessa sovelluksen kanssa. Yleensä Android-sovellus koostuu useasta aktiviteetista ja ne ovat löyhästi sidottuna toisiinsa. Tyypillisessä ohjelmassa on yksi pääaktiviteetti, joka käynnistyy ohjelman käynnistyessä, ja tästä aktiviteetista käynnistetään ohjelman eri toimintoja varten omia aktiviteetteja. (5.)

Jokainen aktiviteetti voi käynnistää uuden aktiviteetin, ja näin tapahtuessa senhetkinen aktiviteetti pysäytetään ja uusi aktiviteetti tulee käyttäjän näkyville. Aktiviteetti voi olla kolmessa eri tilassa, palautettu, tauko ja pysäytetty. Palautetussa tilassa (resumed) aktiviteetti on etualalla ja käyttäjän nähtävissä. Tilasta käytetään myös nimeä ajossa (running). Tauko-tilassa jokin toinen aktiviteetti on etualalla, mutta on vain osittain läpinäkyvä tai ei täytä näytön pinta-alaa koko-

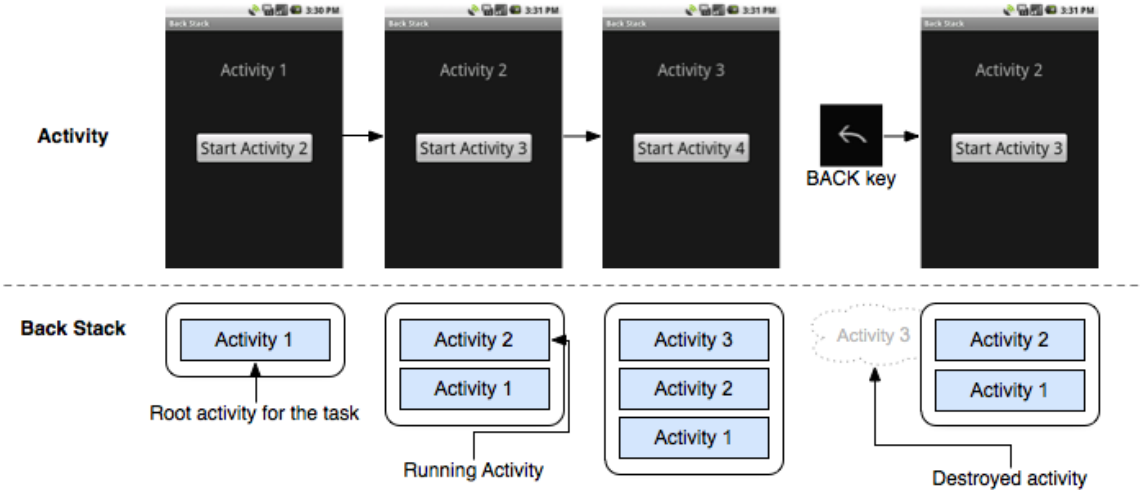
naan. Tässä tilassa käyttöjärjestelmä voi tuhota aktiviteetin, jos käyttöjärjestelmä tarvitsee sen käyttämää muistia. Pysäytetty-tila on tauko-tilan kanssa samankaltainen, mutta tässä tilassa jokin toinen aktiviteetti peittää pysäytetyn aktiviteetin kokonaan. Myös pysäytetyn tilan omaavan aktiviteetin käyttöjärjestelmä voi tuhota halutessaan. Erona pysäytetyllä ja tauko-tilassa olevalla aktiviteetilla on se, että pysäytetty ei ole enää kytkeytyneenä ikkunamanageriin, kun taas tauolla oleva on. Kuvassa 4 on esitetty aktiviteetin elinkaari ja eri tiloissa kutsuttavat callback-metodit. (5.)



KUVA 4. Aktiviteetin elinkaari (5)

Järjestelmässä on käytössä aktiviteettipino (back stack), joka mahdollistaa takaisinpäin palaamisen. Pinossa on käytössä perinteinen LIFO-periaate (Last In, First Out). Uuden aktiviteetin käynnistyessä se lisätään (push) pinoon ja se saa huomion (focus). Kun aktiviteettia ei enää tarvita, se voidaan poistaa (pop) pi-

nosta, jolloin se tuhoutuu ja aikaisempi aktiviteetti tulee käyttöön. Kuvassa 5 havainnollistetaan aktiviteettipinon käyttäytymistä. (6.)



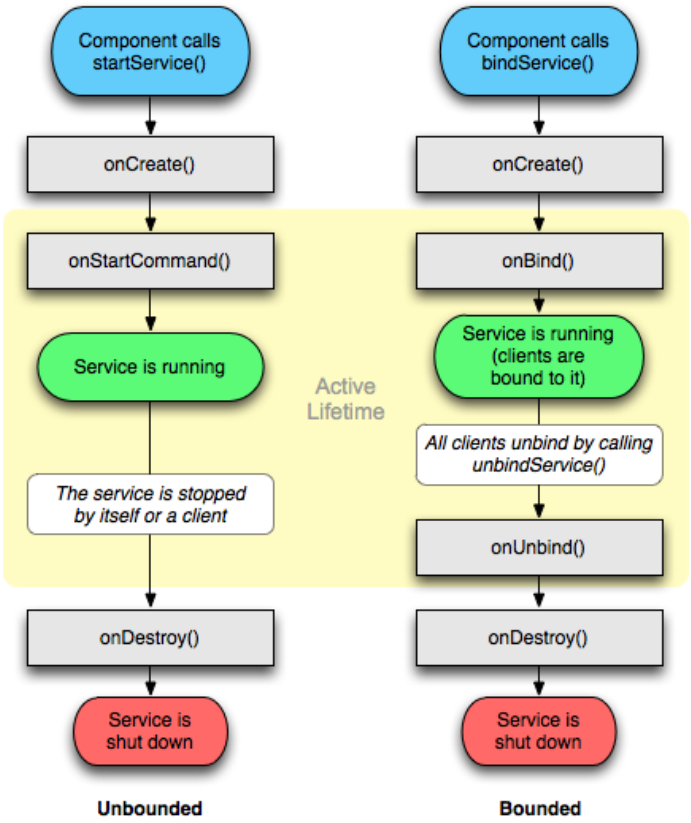
KUVA 5. Aktiviteettipino (6)

Services

Palvelut (Services) on Androidin tapa mahdollistaa pitkäaikaisia operaatioita. Palveluita ajetaan taustalla ja ne eivät sisällä minkäänlaista käyttöliittymää. Ohjelma voi käynnistää palvelun ja se voidaan pitää ajossa, vaikka kyseistä ohjelmaa ei enää käytettäisikään. Palveluita voidaan käyttää esimerkiksi käsittelemään tiettyä verkkoliikennettä tai toistamaan musiikkia. (7.)

Palveluita voi olla kahden tyyppisiä, started ja bound. Palvelu on started silloin, kun ohjelman jokin komponentti (esimerkiksi aktiviteetti) käynnistää palvelun. Kun palvelu on käynnistetty, se pysyy ajossa, vaikka sen käynnistänyt komponentti tuhoutuisikin. Yleensä started-palvelu suorittaa vain yhden toiminnon eikä palauta suoranaisesti sen käynnistäjälle mitään. Esimerkkinä voidaan ottaa palvelu, joka lataa tiedoston verkosta. Kun tehtävä on suoritettu, se pysäyttää itse itsensä. Palvelu on bound, kun komponentit liittyvät siihen kutsumalla bindService-metodia. Palvelu tarjoaa asiakas-palvelinrajapinnan komponentin ja palvelun väliseen kommunikointiin. Rajapinta toimii myös eri prosessien välillä IPC:tä (Interprocess communication) hyväksikäyttäen. Bound-palvelu pysyy ajossa niin kauan, kun jokin komponentti on siihen kytkeytyneenä. Palveluun voi olla kytkettynä useita komponentteja yhtäaikaisesti, mutta kaikkien irtaannuttua palvelu

tuhotaan. Kuvassa 6 käyvät ilmi kummankin tyyppisen palvelun elinkaari ja kutsuttavat callback-metodit. (7.)



KUVA 6. Palvelun elinkaari (7)

BroadcastReceiver

Yleiset vastaanottimet (BroadcastReceiver) ovat komponentteja, jotka vastaavat järjestelmän laajuisiin viesteihin. Suurin osa viesteistä on peräisin käyttöjärjestelmältä, mutta sovellukset voivat myös itse lähettää niitä. Käyttöjärjestelmä lähettää viestejä esimerkiksi silloin, kun akun varaus on vähäinen, tekstiviesti saapuu tai näyttö sammutetaan. (8.)

Vastaanottimia on kahta eri tyyppiä, normaali sekä säännöllinen. Normaali vastaanotin on täysin epäsynkroninen, joten vastaanottimet ajetaan satunnaisessa järjestyksessä. Säännölliset vastaanottimet ajetaan viestin saapuessa aina tietyssä järjestyksessä. Tämä mahdollistaa esimerkiksi viestin peruuttamisen jossakin kohtaa vastaanotinketjua. (8.)

Jotta sovellus voi vastaanottaa viestejä, sen täytyy rekisteröidä oma vastaanotin sekä pyytää tarvittavat oikeudet (8).

Intent

Kolmen edellä esitellyn komponentin välinen viestintä tapahtuu Intenteillä. Intentejä voidaan lähettää sovelluksen sisällä sekä sovelluksien välillä. Intent-objekti sisältää abstraktin kuvauksen suoritettavasta toiminnosta. (9.)

Jokaiselle komponentille on erillinen mekanismi käyttää Intentejä. Aktiviteeteissa Intentin avulla voidaan luoda uusi aktiviteetti, lähettää jo olemassa olevalle aktiviteetille uutta tietoa tai palauttaa tietoa aktiviteetista. (9.)

2.5 Android Market

Android Market on Android-käyttöjärjestelmää käyttävien laitteiden sovelluskauppa. Android Market sekä sitä käyttävä mobiilisovellus (kuva 7) on Googlen kehittämä. Sovelluskauppa on kehittäjille helppo paikka jakaa omia sovelluksiaan, koska suurin osa käyttäjistä on kytkeytynyt siihen. Sovelluskauppaan pääsemiseen tarvitaan vain Googlen tunnukset. (2.)

Sovelluksien lataaminen laitteeseen tapahtuu internetin välityksellä, joko käyttämällä mobiiliverkkoa tai WLAN-yhteyttä. Sovelluskaupassa on tarjolla runsaasti sekä ilmaisia että maksullisia sovelluksia. Maksullisiin sovelluksiin Google tarjoaa lisensointipalvelun, jonka avulla kehittäjät voivat suojata sovelluksiaan laittomalta jakamiselta. (2.)

Sovelluksien saamiseksi sovelluskauppaan tarvitaan kehittäjätunnukset. Kehittäjätunnukset saa itselleen maksamalla 20 dollarin maksun. Tämän jälkeen kehittäjä voi lisätä kauppaan haluamansa määrän sovelluksia. (2.)

Sovelluksien päivittäminen on kehittäjille helppoa sovelluskaupan välityksellä. Kehittäjä julkaisee uuden version sovelluksesta ja käyttäjän laitteessa oleva Android Market -sovellus ilmoittaa automaattisesti uuden version olevan saatavilla. (2.)

KUVA 7. Android Market -sovellus

3 HATTRICK-PELIN RAKENNE

Hattrick on ruotsalaisten kehittämä Internetissä pelattava jalkapallomanageri-peli. Pelin kehittämisen aloitti ExtraLives AB ja ensimmäinen versio julkaistiin elokuussa 1997. Vuonna 2003 Johan Gustafson perusti yrityksen nimeltä Hattrick Limited, joka osti Hattrickin sen alkuperäiseltä kehittäjältä. Tällä hetkellä pelissä on 128 eri maata ja se on käännetty 51:lle eri kielelle. Maailmanlaajuisesti pelaajia on noin 700 000. (11.)

3.1 Pelaaminen

Pelissä pelaajan tarkoituksena on johtaa omaa virtuaalista jalkapallojoukkuetta. Managerin tehtäviin kuuluu esimerkiksi pelaajien ostaminen ja myyminen, kokoonpanojen asettaminen otteluihin, stadionin rakentaminen ja pelaajien harjoittaminen. (12.)

3.2 Sarjajärjestelmä

Jokaisella Hattrickiin lisätyllä maalla on oma sarjansa. Divisioonien eli sarjatasojen määrä vaihtelee maiden välillä kyseisen maan pelaajamäärän mukaan. Kaikissa sarjoissa on kahdeksan joukkuetta. Kauden aikana jokainen joukkue pelaa 14 sarjaottelua eli kaksi kertaa jokaista samaan sarjaan kuuluvaa joukkuetta vastaan. (12.)

3.3 Joukkue

Joukkueet koostuvat virtuaalisista jalkapallon pelaajista. Pelaajia joukkueessa on yleensä 20–30 ja maksimimäärä on 50. Joukkueen omistaa Hattrick-peliä pelaava manageri. Manageri voi antaa joukkueella haluamansa nimen. Hattrick-pelissä on myös niin sanottuja botti-joukkueita, joita ei ohjaa kukaan oikea ihminen. Näitä joukkueita käytetään täydentämään alempia sarjatasoja. (13.)

3.4 Pelaajat

Hattrick-pelissä jokaisella virtuaalisella jalkapallon pelaajalla on nimi, ikä, taidot, persoonallisuus, kunto ja mahdollinen erikoisuus. Pelaajien nimet generoidaan

silloin, kun ne luodaan peliin eikä sitä voida myöhemmin muuttaa. Niminä käytetään oikeita etu- ja sukunimiä, joita on pelaajan syntymämaassa käytössä. (13.)

Hattrick-pelissä pelaajan ikä ilmoitetaan vuosina ja päivinä. Yksi vuosi pelissä kestää 112 päivää. Jotta pelaaja voi liittyä joukkueeseen ja osallistua peleihin, pelaajan täytyy olla 17 vuotta vanha. Maksimi-ikää pelissä ei ole, mutta 30 ikävuoden jälkeen loukkaantumisriski nousee huomattavasti ja loukkaannuttuaan pelaaja ei välttämättä parannu koskaan. (13.)

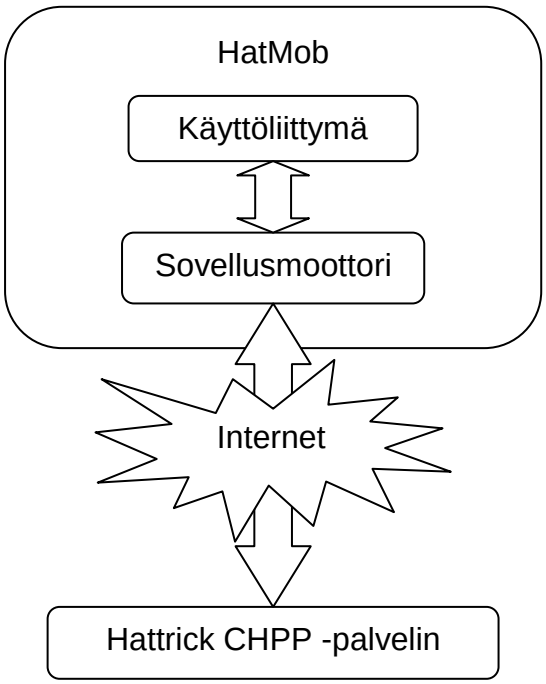
Pelaajan taidot kertovat kuinka hyvä pelaaja on eri osa-alueilla. Taitoja ovat kestävyys, maalivahti, pelinrakennus, syöttö, laituri, puolustus, maalinteko ja erikoistilanteet. Jokainen pelipaikka hyödyntää joitakin näistä taidoista ja mitä paremmat ne ovat sitä paremmin pelaaja pelaa kyseisellä pelipaikalla. Persoonallisuutta pelissä kuvataan neljällä ominaisuudella, joita ovat johtajuus, miellyttävyys, rehellisyys sekä aggressiivisuus. Kunto kertoo pelaajan senhetkisestä pelikunnosta. Huonokuntoisen pelaajan peliesitykset ovat heikommat, kuin ne olisivat hyvällä kunnolla. Pelaajan kuntoon vaikuttaa moni asia, mitä ei ole pelinkehittäjien toimesta kerrottu. (13.)

4 HATMOB-ASIAKASSOVELLUS

HatMob on Androidille kehitetty sovellus, joka tarjoaa työkaluja Hattrickin pelaamista varten mobiililaitteeseen. Alkuperäisessä suunnitelmassa oli tarkoituksena toteuttaa sovellukseen vain mahdollisuus seurata Hattrickissä pelattavia otteluita suorana ilman verkkoselainta. Hyvin pian kehityksen alettua HatMob-sovelluksesta kuitenkin haluttiin tehdä kattavampi työkalu Hattrick-pelin pelaamiseen. Mukaan tulleita ominaisuuksia ovat joukkueen perustiedot, sarjataulukko, tulevat ottelut sekä pelaajat.

4.1 Sovelluksen rakenne

Sovellus voidaan jakaa kahteen osaan, käyttöliittymään ja sovellusmoottoriin. Käyttöliittymän tehtävänä on näyttää käyttäjälle tietoja ja vastata käyttäjän tekemiin syötteisiin, kun taas sovellusmoottorin tehtäviin kuuluu tietojen lataaminen verkosta sekä niiden jäsentäminen ja tallentaminen. Kuvassa 8 pyritään selkeyttämään sovelluksen rakennetta sekä tiedon kulkua palvelimelta sovellukseen.



KUVA 8. HatMob-sovelluksen perusrakenne

Sovellusmoottorissa ja koko sovelluksessa käytetään tarkkailija-mallia (observer pattern) olioiden välisten päivitysriippuvuuksien rakentamiseen.

4.2 Käyttöliittymä

Sovelluksen käyttöliittymä pyrittiin suunnittelemaan mahdollisimman yksinkertaiseksi ja nopeaksi käyttää. Käyttöliittymässä päädyttiin käyttämään niin sanottua välilehtimallia, jossa jokainen välilehti sisältää oman informaationsa (kuva 9). Ottelunäkymälle kuitenkin lisättiin oma aktiviteettinsa, jotta näytölle saatiin mahtumaan mahdollisimman monta ottelutapahtumaa yhtäaikaisesti.

KUVA 9. Sovelluksen päänäkymä

Seuraavaksi käydään läpi sovelluksen eri näkymien tarkoitus sekä niissä esitettävät tiedot.

MyHT

Sovelluksen käynnistyessä MyHT on ensimmäinen näkymä, jonka käyttäjä näkee. Näkymässä esitetään joukkueen perustietoja, jotka ovat yleiskatsaus, areena, henkilökunta ja kannattajat. Yleiskatsauksessa käy ilmi joukkueen nimi, sijoitus sarjassa, managerin nimi sekä sijoitus kaikkien samasta maasta olevien joukkueiden välillä. Arena-kohdasta käyttäjä näkee oman areenansa nimen,

paikkakunnan sekä katsomon koon. Joukkueen henkilökunnan määrää sekä siitä tulevia kustannuksia voi tarkastella henkilökunta-kohdasta. Viimeisenä kohtana näkymässä on fanien tiedot. Siitä käyvät ilmi fanien mieliala, määrä sekä odotukset tulevalle kaudelle.

HT Live!

Tässä näkymässä käyttäjä näkee reaaliaikaisessa seurannassa olevat ottelut. Jokaisesta ottelusta käyvät ilmi pelaavat joukkueet, ottelun tyyppi sekä tilanne tai alkamisajankohta, jos ottelu ei ole vielä alkanut. Näkymästä pääsee siirtymään jokaisen ottelun tarkempaan näkymään valitsemalla jonkin otteluista.

Matches

Ottelut-näkymästä käyttäjä voi katsella menneiden otteluiden tuloksia, nähdä tulevat ottelunsa sekä lisätä näitä reaaliaikaiseen seurantaan. Menneistä otteluista ovat esillä pelanneet joukkueet, ottelun tyyppi, ottelun aloitusaika sekä lopputulos.

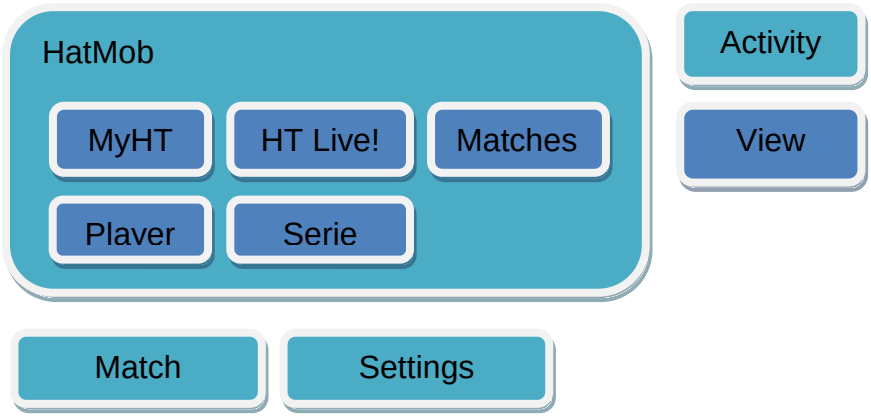
Serie

Sarja-näkymässä esitetään joukkueen sarjan senhetkinen tilanne. Taulukosta käyvät ilmi joukkueen sijoitus, pisteet, tehdyt ja päästetyt maalit, maalien erotus sekä sarjapisteet.

Settings

Settings-näkymästä käyttäjä pääsee muokkaamaan sovelluksen asetuksia. Asetuksista löytyvät mahdollisuudet vaikuttaa suorana seurattavien otteluiden päivitystiheyteen sekä näytettävän ajan muotoon.

Kuvassa kymmenen esitetään sovelluksen näkymärakenne. Näkymärakenteesta ilmenee sovelluksen aktiviteetit sekä aktiviteettien sisältämät näkymät (kuva 10).

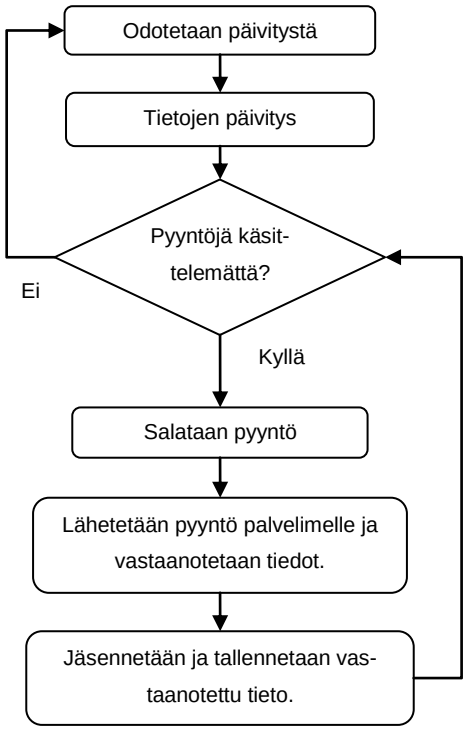


KUVA 10. Näkymärakenne

4.3 Sovellusmoottori

Sovellusmoottorin tehtäviin kuuluu tietojen hakeminen palvelimelta, jäsentäminen sekä tallentaminen.

Sovelluksen sujuvan käytön takaamiseksi sovellusmoottoria ajetaan omassa säikeessä. Tällä tavoin tietojenkäsittelyssä kuluva aika ja sen aiheuttama kuorma suorittimelle eivät huononna käyttäjäkokemusta turhilla odotuksilla. Saatuaan uutta tietoa palvelimelta sovellusmoottori ilmoittaa tiedosta kiinnostuneelle oliolle. Kuvassa 11 esitetään sovellusmoottorin toimintaa kuvaava tilakaavio.



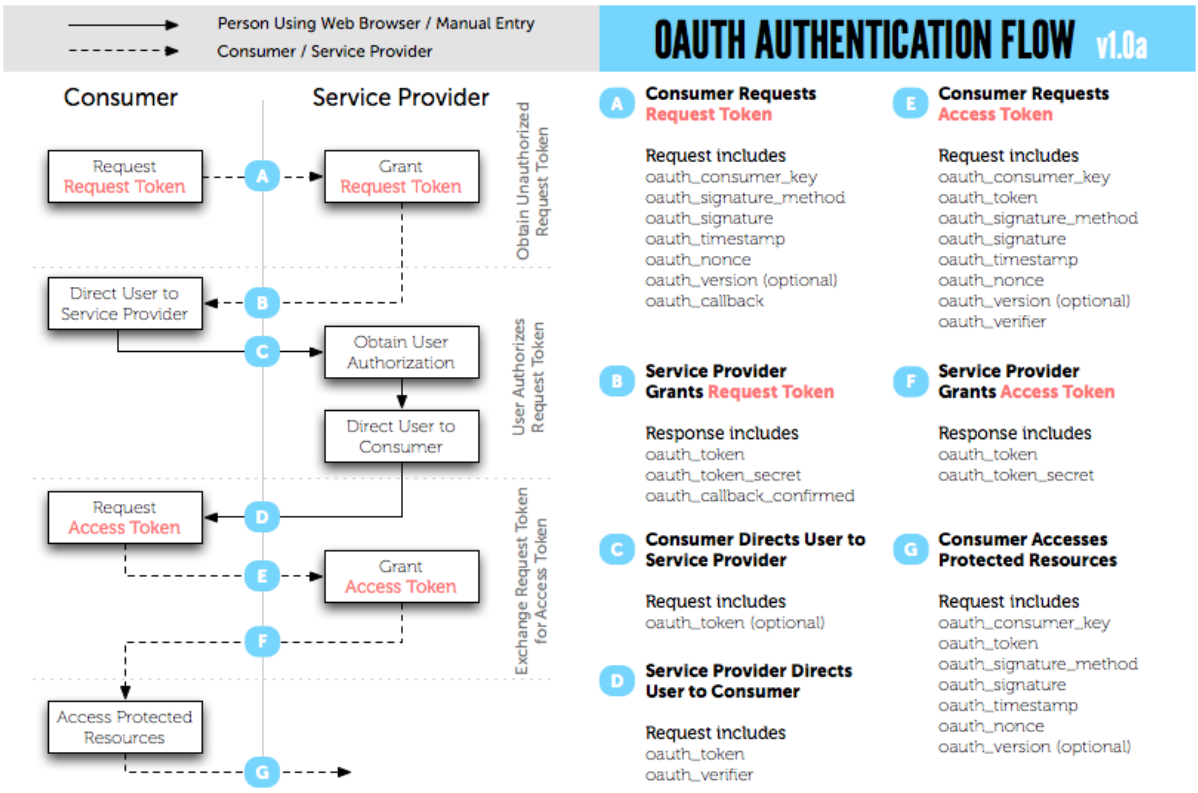
KUVA 11. Pelkistetty tilakaavio sovellusmoottorin toiminnasta

Käyttäjän todentaminen

OAuth-protokolla on suunniteltu sovelluksien rajapintojen suojaukseen. Tunnistautumisen toimintaperiaate eroaa tavanomaisesta tunnistautumisesta sillä, että siinä on aina kolme osapuolta. Ensimmäisenä osapuolena on palvelu, joka haluaa tarjota rajapinnan palvelun käyttämiseen ulkopuolelta. Toisena osapuolena on sovellus, jonka tarvitsee päästä käsiksi käyttäjän tietoihin palvelussa. Viimeisenä osapuolena on käyttäjä, joka haluaa käyttää sovellusta ja antaa näin ollen sovellukselle oikeuden palvelussa sijaitseviin tietoihin. OAuth-protokollan käytön hyötyinä voidaan pitää sitä, että käyttäjän ei tarvitse jakaa palvelun tunnuksia sovelluksille, jotka voisivat väärinkäyttää niitä. Palveluun ei myöskään tarvitse kehittää erillisiä tunnuksia rajapinnan käyttöä varten. (10.)

Seuraavaksi käydään läpi yksinkertaistetusti tunnistautumisen kulku. Tunnistautuminen alkaa käyttäjän halusta käyttää palvelua sovelluksen avulla. Sovellus kysyy palvelulta, missä verkko-osoitteessa käyttäjä voi kyseisen luvan antaa. Sovellus ohjaa käyttäjän palvelun osoittamalle sivustolle, jossa käyttäjä vielä hyväksyy sovelluksen haluamat oikeudet ja kirjautuu sivustolle palvelun tunnuksilla. Käyttäjän kirjauduttua hänet ohjataan takaisin sovellukseen mukanaan

tarvittavat tiedot tunnistautumisen loppuun viemiseen. Tunnistautumisen onnistuttua sovelluksella on hallussaan tarpeelliset tiedot pyyntöjen salaamiseen sekä niiden suorittamiseen palvelusta. Myöhemmin käyttäjä voi halutessaan perua sovelluksella olevat oikeudet käytettävän palvelun kautta. Tarkempi kuvaus tunnistaumisen kulusta on esitetty kuvassa 12. (10.)



KUVA 12. OAuth-tunnistautumisen kulku (10)

Tiedonsiirto

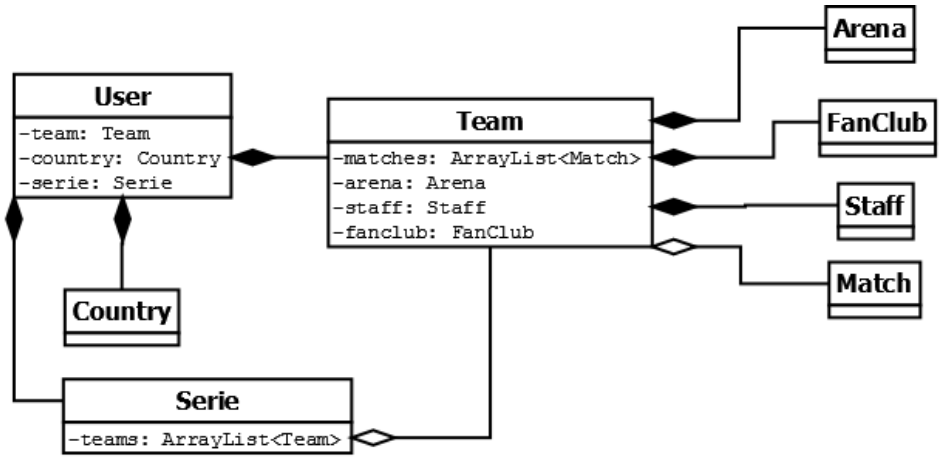
Sovelluksen ja palvelimen välinen tiedonsiirto on toteutettu HTTP-protokollalla. Jokainen palvelimelle lähetettävä GET-pyyntö salataan OAuth-protokollaa käyttäen. Vastauksena onnistuneeseen pyyntöön palvelin lähettää sovellukselle sen pyytämän tiedon XML-tiedostona.

Tietojen jäsentäminen

Tietojen jäsentäminen on yksi sovellusmoottorin tärkeimmistä tehtävistä. Jäsentämisellä tarkoitetaan tässä tapauksessa ladattujen XML-tiedostojen tulkintaa sekä niistä oleellisen tiedon tallentamista. XML-tiedostojen jäsentämiseen on olemassa useita menetelmiä. Android-alustalla oletuksena tuettuja menetelmiä

ovat SAX, DOM ja Pull. Näistä menetelmistä sovellus hyödyntää kahta jälkimmäistä. Menetelmien välillä on eroja nopeudessa sekä käytön helppoudessa. Parasta menetelmää ei pystytä sanomaan, koska esimerkiksi nopeus riippuu jäsennettävän tiedon rakenteesta ja siitä, mitä osia tiedosta halutaan tulkita.

Sovelluksessa tietojen jäsentäminen tapahtuu heti lataamisen jälkeen sovellusmoottorissa. Halutut tiedot tallennetaan dataluokkiin, joiden rakenne esitetään kuvassa 13. Kuvan pitämiseksi yksinkertaisena siitä on jätetty pois suurin osa luokkien datajäsenistä.



KUVA 13. Dataluokkien rakenne

5 KEHITETYT OMINAISUUDET

Tässä luvussa käydään läpi työn aikana sovellukseen kehitetyt ominaisuudet sekä käyttöliittymän optimointia.

5.1 Tietokanta

Uusien ominaisuuksien myötä sovellus noutaa palvelimelta enemmän ja enemmän tietoa, minkä seurauksena tietojen lataaminen ja jäsentäminen vie enemmän aikaa. Tavoitteena tietokannan toteuttamisella oli käyttökokemuksen parantaminen. Tietokannasta tietoja on nopea noutaa ja esittää käyttäjälle jo heti sovelluksen käynnistyttyä. Aiemmin tietoja alettiin ladata palvelimelta, kun sovellus käynnistyi. Tietojen lataaminen aiheutti suurta viivettä, ennen kuin käyttäjä näki oman joukkueensa tietoja. Toisena tarkoituksena tietokannalla on myöhemmin toteutettava mahdollisuus kerätä esimerkiksi pelaajien taitojen muutoksia ja esittää näitä käyttäjälle. Tällä tavoin käyttäjä pystyisi seuraamaan pelaajien kehitystä sekä selaamaan heidän historiaa.

Sovelluksessa käytetään SQLite-tietokantaa, jonka käyttämiseen Android-alustasta löytyvät apuluokat valmiina. SQLite on relaatiotietokantajärjestelmä, jota käytetään yleisesti useilla eri alustoilla.

Tietokannan käsittelyä varten sovellusmoottoriin kehitettiin oma luokka, joka sisältää lukuisia metodeja hoitamaan eri tietojen lisäämistä, päivittämistä, noutamista sekä poistamista tietokannasta.

5.2 Pelaajanäkymä

Koska pelaajat ovat Hattrick-pelin hyvin oleellinen osa, päätettiin sovellukseen suunnitella ja toteuttaa heidän esittämisen mahdollistava näkymä. Tämä osio työstä alkoi luonnollisesti suunnittelemalla ominaisuutta kokonaisuutena, menemättä vielä tarkempiin yksityiskohtiin. Suunnittelun myötä kävi ilmi, että esitettävää tietoa yhtä pelaaja kohti on paljon, ja että pelaajien kuvien kokoaminen vaatisi paljon työtä. Suunnittelun jälkeen siirryttiin toteuttamaan sovellusmoottoriin tarvittavat muutokset, jotta se kykenisi tarjoamaan pelaajista tarpeelliset tiedot. Seuraavaksi oli luvassa näkymän käyttöliittymän suunnittelu ja toteutus se-

kä sen integrointi sovellusmoottoriin. Viimeiseksi vaiheeksi jäi pelaajien kuvien latauksen, kokoamisen ja tallentamisen suunnittelu sekä toteutus. Seuraavaksi käydään tarkemmin läpi kaikki edellä mainitut työvaiheet.

Jäsentäminen

Pelaajanäkymää varten sovellusmoottoriin piti tehdä tuki kahdelle uudelle XML-tiedostolle. Toisessa tiedostossa kuvataan kaikkien joukkueen pelaajien tiedot ja toisessa pelaajien kuvia koskevat tiedot.

Molempien tiedostojen jäsentämiseen käytettiin Pull-menetelmää. Se oli hyvin helppoa aiemmin toteutettujen rajapintojen ja apuluokkien avulla. Jäsennetyt tiedot tallennettiin tietokantaan myöhempää käyttöä varten.

Käyttöliittymä

Käyttöliittymää suunniteltaessa esillä kävi monenlaisia vaihtoehtoja pelaajatietojen esittämiseen, mutta lopuksi päädyttiin jo muuallakin sovelluksessa käytettyyn laajennettavan listan käyttämiseen. Käyttöliittymän suunnittelun ongelmana oli näytettävän tiedon runsaus, jonka johdosta osa tiedoista piilotettiin napin taakse. Nappia painamalla piilotetut tiedot laajenevat kuvassa 14 jo näkyvien tietojen väliin.

KUVA 14. Pelaajanäkymä

Pelaajanäkymä koostuu ylhäällä vasemmalla olevasta pelaajan kuvasta, sen oikealla puolella olevista tiedoista sekä alhaalla esitetyistä pelaajan taidoista. Kaikki pelaajaa koskevat tiedot ovat esitetty tekstinä sekä niitä vastaavina numeroarvoina.

Pelaajien kuvat

Hattrick-pelissä pelaajien kuvat koostuvat osista, joten jokaiselle pelaajalle on helppo generoida omanlaisensa ulkonäkö. Osilla tarkoitetaan esimerkiksi silmiä, nenää, hiuksia, pään muotoa ja niin edelleen. Myös CHPP-sovelluksille tarjotaan vain tiedot pelaajan kuvassa esiintyvistä osista ja niiden perusteella kuva voidaan koota ja esittää.

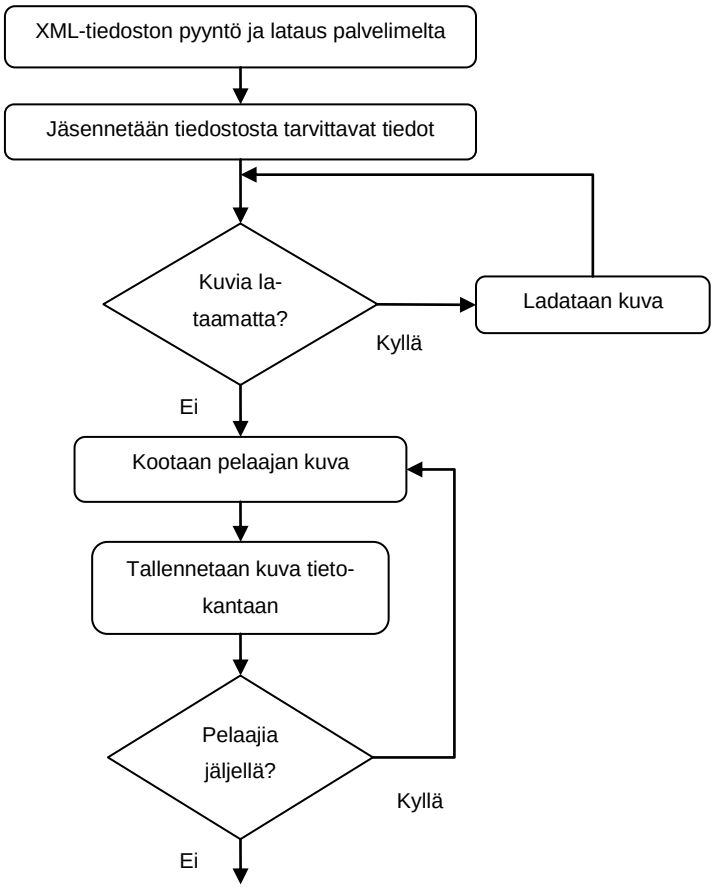
Kuvien lataaminen suunniteltiin ja toteutettiin niin, että samoja osia ei ladata useaan kertaan sekä jokainen osa tallennetaan tietokantaan, josta niitä voidaan käyttää myöhemmin uudelleen. Jokainen koottu kuva myös tallennetaan pelaa-

jan tietoihin, jolloin sen piirtäminen ja lataaminen on nopeampaa, koska sitä ei tarvitse aina koota osista uudestaan.

Ensimmäisenä sovellusmoottoriin piti toteuttaa tuki uudelle XML-tiedostolle. XML-tiedostosta ilmenee tarvittavat tiedot kuvien kokoamiseen. Tiedoissa on käytettävä taustakuva, jokaisen osan XY-koordinaatit sekä osoite, mistä osan kuva voidaan ladata. XML-tiedostoa jäsennettäessä kerätään talteen kyseiset tiedot taulukoihin. Yhteen taulukkoon kerätään tiedot, mitä osaa kukin pelaaja käyttää, ja toiseen taulukkoon kaikki tarvittava osat. Jäsentämisen jälkeen osa-taulukko käydään läpi ja sieltä poistetaan useaan kertaan esiintyvät kuvan osat.

Seuraavaksi sovellusmoottoriin toteutettiin tuki kuvien lataamiseen. Se tapahtuu myös HTTP-protokollaa käyttäen. Kuvien lataukseen kehitetty luokka toteuttaa Runnable-rajapinnan. Tämä tarkoittaa sitä, että kuvien lataaminen suoritetaan omassa säikeessä, eikä se tällä tavoin aiheuta odotteluja tai hidastumista sovelluksen käytössä. Jokainen ladattu kuva tallennetaan sellaisenaan tietokantaan, josta sitä voidaan käyttää myöhemmin.

Kun kaikki kuvat on saatu onnistuneesti ladattua palvelimelta tietokantaan, aloitetaan kuvien kokoaminen jokaiselle pelaajalle. Kuvien kokoaminen aloitetaan hakemalla senhetkisen pelaajan kuvan tiedot tietokannasta. Tämän jälkeen luodaan uusi tyhjä kuva, johon piirretään jokainen pelaajan kuvan osa koordinaattien osoittamalle paikalle sekä skaalataan lopullinen kuva halutun kokoiseksi. Lopuksi valmis kuva tallennetaan kyseisen pelaajan tietoihin tietokantaan. Nämä toimenpiteet suoritetaan luonnollisesti kaikille pelaajille. Kun kaikki pelaajat on käyty läpi ja niiden kuvat koottu, ilmoitetaan pelaajanäkymälle, jotta se tietää piirtää päivittyneet kuvat. Kuvassa 15 esitetään kuvien luonnin tilakaavio.



KUVA 15. Pelaajien kuvien luonnin tilakaavio

5.3 Käyttöliittymän optimointi

Käyttäjäkokemus on tärkeä asia kehitystiimille. Siksi sovelluksen käyttöliittymää päätettiin optimoida. Ongelmakohtina käyttöliittymässä oli listanäkymien hidastelu niitä selattaessa. Tätä osaa aloitettaessa kenelläkään tiimistä ei ollut tietoa, mistä hidastelut johtuivat tai mitä niille olisi tehtävissä. Tämän johdosta työ koostui suurelta osaltaan tiedon etsimisestä Internetistä sekä sovelluksen rakenteen tutkimisesta. Tässä kohdassa käydään läpi löydetyt ongelmat ja niihin löydetyt ratkaisut.

Käytettäessä Android-alustan tarjoamaa listanäkymää jokaisen lapsinäkymän piirtämistä varten järjestelmä kutsuu getView-metodia. Jos kehittäjä haluaa käyttää omaa asetteluaan lapsinäkymälle, hänen täytyy ylikirjoittaa tämä metodi, jotta se kykenee tarjoamaan halutunlaiset tiedot järjestelmälle. Jo varhaisessa vaiheessa ongelmakohtia etsiessä huomattiin getView-metodissa kuluvan suoritusajan olevan liian suuri. Tämän ongelman ratkaisuksi näkymien luokkiin

toteutettiin niin sanotut pidikeluokat. Nämä luokat sisältävät oliot kaikkiin kyseisen listan lapsinäkymän komponentteihin. Tiedot olioihin etsitään, kun näkymää piirretään ensimmäistä kertaa, jonka jälkeen niitä voidaan käyttää aina uudestaan. Tämä parannus vähensi getView-metodin suoritusaikaa huomattavasti.

Edellisen parannuksen ansiosta listanäkymien käytettävyys parani, mutta suorituskykyyn ei oltu vieläkään tyytyväisiä. Seuraavaksi alettiin tutkia mahdollisuuksia vähentää getView-metodin kutsujen määrää, koska määrän huomattiin olevan suuri lapsinäkymien lukumäärään nähden. Ainoaksi keinoksi kutsujen vähentämiseen löydettiin asettelutiedostojen yksinkertaistaminen ja parantaminen. Asettelutiedostoja muokattiin käyttämään mahdollisimman matalan tason komponentteja. Komponenttien kokomäärityksiä säädettiin niin, että järjestelmä joutuu mahdollisimman vähän suorittamaan laskentaa komponentteja piirrettäessä. Edellä mainittujen parannuksien johdosta käyttöliittymän käytettävyys parani huomattavasti.

6 YHTEENVETO

Työssä oli tarkoituksena suunnitella ja toteuttaa HatMob-sovellukseen mahdollisuus selata käyttäjän pelaajia, integroida tietokanta sekä optimoida käyttöliittymää. Työn suorittamisessa ei suuria ongelmia tullut vastaan ja kaikki siihen suunnitellut tehtävät saatiin suoritettua. Haastavimpana tehtävänä voidaan pitää käyttöliittymän optimointia, joka vaati paljon asiaan perehtymistä ja vei paljon aikaa.

Työtä tehdessä karttui paljon lisää kokemusta sovelluskehityksestä Android-käyttöjärjestelmälle sekä sovelluskehityksestä yleisesti. Työn aikana sekä aikaisemmin projektissa eniten hankaluuksia on aiheuttanut Hattrick-pelin tarjoaman rajapinnan paikoittain epämääräinen toiminta.

Sovelluksen kehitystä on tarkoitus jatkaa vielä tämän työn jälkeen. Suunnitelmissa sovellukseen on tulossa lukuisia uusia ominaisuuksia, kuten pelaajien kehityksen seuraaminen, otteluraporttien esittäminen ja kokoonpanojen asettaminen otteluihin.

LÄHTEET

1. Brief History Of the Android OS. 2011. Saatavissa: http://www.brighthub.com/mobile/google-android/articles/18260.aspx. Hakupäivä 12.12.2011.

2. Android. 2011. Saatavissa: http://en.wikipedia.org/wiki/Android_(operating_system). Hakupäivä 1.11.2011.

3. What is Android?. 2011. Saatavissa: http://developer.android.com/guide/basics/what-is-android.html. Hakupäivä 12.12.2011.

4. Android software development. 2011. Saatavissa: http://en.wikipedia.org/wiki/Android_software_development. Hakupäivä 12.12.2011.

5. Activities. 2011. Saatavissa: http://developer.android.com/guide/topics/fundamentals/activities.html. Hakupäivä 12.12.2011.

6. Tasks and Back Stack. 2011. Saatavissa: http://developer.android.com/guide/topics/fundamentals/tasks-and-back-stack.html. Hakupäivä 12.12.2011.

7. Services. 2011. Saatavissa: http://developer.android.com/guide/topics/fundamentals/services.html. Hakupäivä 12.12.2011.

8. BroadcastReceiver. 2011. Saatavissa: http://developer.android.com/reference/android/content/BroadcastReceiver.html. Hakupäivä 12.12.2011.

9. Intents and Intent Filters. 2011. Saatavissa: http://developer.android.com/guide/topics/intents/intents-filters.html. Hakupäivä 12.12.2011.

10. OAuth Core 1.0. 2011. Saatavissa: http://oauth.net/core/1.0/. Hakupäivä 12.12.2011.

11. Hattrick. 2011. Saatavissa: http://wiki.hattrick.org/wiki/Hattrick. Hakupäivä 2.11.2011.

12. Hattrick. 2011. Saatavissa: http://en.wikipedia.org/wiki/Hattrick. Hakupäivä 12.12.2011.

13. Players. 2011. Saatavissa: http://wiki.hattrick.org/wiki/Players. Hakupäivä 12.12.2011.